**Value:** is smalles unit of information

**values can  be stored in variables**

**variable name conventions**

camel casing is used in javascript(convention)

$ and \_ are the spl symbols which can be used to name the variables

**Variable name rules**

variable **should not start with number**

**keywords should not be used as variable names**

**function cannot be used as variable name**

name also is reserved keyword

dont start variable name with upperCase

for constants all letters must be caps(eg:PI)

**Note:**every value has an primitive data type(it is the value which has datatype not the variable)

**number** used for decimals and integers

**string**

**boolean**

**undefined**:value taken by a variable that is not yet defined

eg let children;

**NULL** also means empty value

**Symbol**(ES2015):Value that is unique and cannot be changed[not useful for now

**BigInt**

Javascript has **dynamic typing**:we do not have to manually define the data type of value stored in a a variable

data types are determined automatically

value has data type not variable

Let,Const,Var:

Eg:

let year;

both value and typeof value is undefined

console.log(year);

console.log(typeof year);

null is not an object but typeof returns object for typeof null

console.log(typeof null);

**let** can be **used to declare variables that change their values later**

**const** can **be used to declare variable that remains constant throughout the entire program**

**var** is oldway of defining variables prior to es6

**let** is block scoped

**var** is function scoped

 let and const **respect block scope**, meaning they are only **accessible within the block where they are defined**.

 **var respects function scope**, **meaning it is accessible anywhere within the function where it is defined but not outside the function**.

Operators(+,-,\*,\*\*,++,--,/,%)

Comparison operators(>,<,>=,<=,===,!==)

**Template literals(` `):**

any expression(i.e anything that holds a value) can be put inside ${}

String with multiple lines is easier using template literal

console.log("String with \nmultiple\nlines");

console.log(`String

multiple

lines`);

control structures(if else)

eg: if (age >= 18) {

  console.log(`Sarah can start driving license 🚗`);

} else {

  const yearLeft = 18 - age;

  console.log(`Sarah is too young.Wait another ${yearLeft} years`);

}

**Type conversion and cohersion**

**type conversion(explicitly by the user):**

**Number()**

**String()**

console.log(Number(inputYear) + 18);

//converting String which cannot be converted to number

console.log(Number("Jonas")); //returns value of NaN implies not a number

console.log(typeof NaN); //returns Number(JS is weird)

console.log(String(23));

**type coercion(implicit conversion):**

// whenever there is a plus operation between Number and String **then js converts number to String**

console.log("I am " + 23 + " years old");

// whenever there is a minus operation between Number and String then js converts String to number

// console.log("23" - "10" - 3);

// console.log("23" / "2");

// let n = "1" + 1; //11

// n = n - 1; //10

// console.log(5 + "2"); //52

// console.log(n);

// console.log(2 + 3 + 4 + "5"); //95

// console.log("10" - "4" - "3" - 2 + "5"); //15

Truthy and falsy values:

5 falsy values: 0,''’,undefined,null,NaN

**Equality operators:**

=== strict equality(doesnt do type coercion)

==   loose equality(does type coercion)

=== checks both value and datatype

== checks only the value

**Note:** strict equality is preffered

**To store value of prompt in a variable:**

const favourite = Number(**prompt**("what's your favourite number"));

//output of prompt is usually string so we need to convert it

Boolean logic(!, &&, ||)

Eg:

const hasDriversLicense = true; //A

const hasGoodVision = true; //B

console.log(hasDriversLicense && hasGoodVision);

console.log(hasDriversLicense || hasGoodVision);

const shouldDrive = hasDriversLicense && hasGoodVision;

if (shouldDrive) {

  console.log(`Sarah is able to drive`);

} else {

  console.log(`Someone else should drive`);

}

Note:break is necessary for each case and also for the deafault

switch (day) {

  case "monday": //day===monnday

    console.log("plan my course structure");

    console.log("goto coding meetup");

    break;

  case "tuesday":

    console.log("Prepare theory videos");

    break;

  case "wednesday":

  case "thursday":

    console.log("write code examples");

    break;

  case "friday":

    console.log("record videos");

    break;

  case "saturday":

  case "sunday":

    console.log("enjoy the weekend");

    break;

  default:

    console.log("not a valid day");

    break;

}

Ternary operator:

const bill = 275;

const tip = bill >= 50 && bill <= 300 ? bill \* 0.15 : bill \* 0.2;

console.log(`the tip value is ${tip}`);

Fundamental Part 2

**"use strict";** activates strict mode which makes easier for developers to avoid accidental errors

strict mode **forbids us to do certain things**

**makes visble the errors in console**

**function:** is a piece of code which we can use again and again

**function declaration syntax**

 function function\_name(parameter1,parameter2)

{

}

**calling function**

function\_name();

eg:logger();

**function expression:**

basicaly what we do here is

we write the function in same way as before

but we dont give it a name

instead we equate it to a variable

and using that variable name function can be called

syntax: const fn\_name=function(parameter1,pa2)

{

return ;

}

**arrow functions(spl type of function expression)**

syntax: const fn\_name=(parameter1,parameter2)=>

{ .....

  return value

}

**Point to remember**: after return the function will  immediately exit

Summary of Functions:

all fns work in the sameway

receive input data

transform data

output data

parameter are placeholder..they hold the data which we pass as arguments

Arrays:

**const friends = ["Michael", "Steven", "Peter"];**

**const y = new Array(1991, 1984, 2008, 2020);**

only primitive variables are immutable if declared as constant

array is not primitive hence it can be mutated even if it is declared as const

but we cannot replace the entire array

eg:friends=['rovin','jasd','sadasdas']

**array can consist of any data type including variables,arrays,expression**

const jonas = [firstname, "Schemdtmann", 2037 - 1991, "teacher", friends];

1 push- **adds element to the end of the array** and also returns the length of new array

2 unshift-**adds element to the start of the array** and also returns the length of new array

3 pop -remove elements from the end and returns the removed element

shift removes the first element(similar to left shift)

**indexOf**-to get position of certain element in array

**includes** - return if that element is in the array or not uses check equality for the check

**Introduction to objects**

**i)key value pairs**

**ii)** **instead of [] bracs in arrays we use { } in objects**

**object literal syntax:**

const jonas = {

  firstname: "robinson",

  lastname: "manuvel",

  age: 2037 - 1991,

  job: "teacher",

  friends: ["michael", "peter", "Steven"],

};

**Dot notation** to retrieve the value of property of object

console.log(jonas.lastname);

**when we have to compute the property name**..**only bracket notation** can be used

const interestedIn = prompt(

  "What do you want to know about jonas?Choose between firstName,lastName,age,job,and friends"

);

if (jonas[interestedIn]) {

  //if value exists print it

  console.log(jonas[interestedIn]);

} else {

  console.log(

    "Wrong request! Choose between firstname,lastname,age,job,and friends"

  );

}

**adding new properties to objects**

jonas.location = "portugual";//using dot notation

jonas["twitter"] = "@jonasschmedtmann";//using bracket notation

console.log(jonas);

note:any function that is attached to an object is called method

**using this keyword**

const jonas = {

  firstname: "robinson",

  lastname: "manuvel",

  birthYear: 1991,

  hasDriversLicence: false,

  job: "teacher",

  friends: ["michael", "peter", "Steven"],

  calcage: function () {

    this.age = 2037 - this.birthYear; //adding new property to currently calling object

    return this.age;

  },

  getsummary: function () {

    this.summary = `Jonas is ${this.calcage()} old teacher,and he ${

      this.hasDriversLicence

        ? `has a driver's license`

        : `Does not have a driver's license`

    }`;

    return this.summary;

  },

};

**GK**

**arrays are objects thats y they have methods**

**jonas.push('sadsdsad'); the array name jonas is object**

**and it has method like push,shift,unshift and so on**

**For loop**

const types = []; //syntax to create empty arrays

for (let i = 0; i < jonasArray.length; i++) {

  //filing types array

  types.push(typeof jonas[i]);

}

**continue and break**

**coninue wiil exit the current iteration and go to next**

for (let i = 0; i < jonasArray.length; i++) {

  if (typeof jonasArray[i] !== "string") {

    continue;

  }

  console.log(jonasArray[i]);

}

**break will completely terminate the loop**

for (let i = 0; i < jonasArray.length; i++) {

  if (typeof jonasArray[i] === "number") {

    break;

  }

  console.log(jonasArray[i]);

}

**While loop**

**it will run while condition is true**

**when we dont have in hand how many times loops will run**

let rep = 1; //initialization

while (rep <= 10) {

  console.log(`lifiting weights repitition ${rep} 🏋️`);

  rep++;//increment

}